'Requirements draaien om van een probleem'

Requirements vormen een moeilijk terrein voor menig ingenieur; de oorzaak daarvan is hun abstracte karakter. Het gaat namelijk niet om het oplossen van problemen, maar om het specificeren ervan. Volgens adviseur James Robertson vormen ze een noodzakelijk vakgebied. Een gesprek over requirements, de specificatie en de herbruikbaarheid ervan.

HANS VAN THIEL

James Robertson houdt zich de laatste jaren bezig met de ontwikkeling van objectgeoriënteerde systemen, technologieoverdrachttechnieken en het determineren van system requirements. Robertson is co-auteur van een boek over requirements[1]. Hij adviseert bedrijven bij het specificeren van requirements en geeft er cursussen over. Robertson is lid van het 'Atlantic Systems Guild'[2], een groep onafhankelijke consulenten op het gebied van software-engineering. Peter Hruschka[3] is ook lid van deze groep.

PT Embedded Systems: Is er verschil tussen het door Hruschka en Hatley ontwikkelde 'Process for System Architecture and Requirements Engineering' en het 'Volere' proces dat u onder de aandacht brengt?

James Robertson: In principe is er geen verschil, maar Hruschka maakt veel gebruik van toestandsdiagrammen. De systemen waar ik meestal mee werk, zijn meer datageoriënteerd. Het voorbeeld uit mijn boek, een systeem om winterse ijsvorming op verkeerswegen te voorspellen en vervolgens tegen te gaan, leunt zwaar op gegevensopslag. In de meeste embedded systemen is dat echter minder belangrijk.

PT Embedded Systems: Waarom vinden engineers de specificatie van requirements zo moeilijk?

James Robertson: Engineers zijn heel goed in het oplossen van problemen en ze werken graag met concrete zaken. Requirements zijn abstract; de beste requirements staan zelfs los van welke implementatie dan ook. Bij requirements gaat het om het specificeren van het probleem, om datgene wat er moet gebeuren, en niet om de oplossing. Veel programmeurs en engineers hebben daar moeite mee.

PT Embedded Systems: U benadrukt in uw boek dat de juiste mensen bij het verzamelen van requirements moeten worden betrokken.

James Robertson: Je moet de belanghebbenden erbij betrekken om ervoor te zorgen dat je geen requirements mist, en om er zeker van te zijn dat de requirements die je specificeert later door de organisatie ondersteund zullen worden. Ook is dit handig om erachter te komen wat de requirements eigenlijk zijn.
Het is meestal niet genoeg om belanghebbenden (Eng. stakeholders) alleen maar te vragen wat ze willen. Het gaat er namelijk om iets uit te vinden waar mensen niet met zoveel woorden om vragen. Denk hierbij bijvoorbeeld aan mobiele telefonie of het Internet. Ingenieurs zijn heel goed in uitvinden; de kunst is echter om ze dingen te laten uitvinden die te verkopen zijn.
Bij requirements gaat het overigens niet alleen om datgene wat mensen nodig hebben, maar ook om wat ze bijvoorbeeld mooi vinden of wat ze bevalt. Als mensen een product of een dienst niet prettig vinden, dan zullen ze er ook geen gebruik van maken. Enkele jaren geleden, bijvoorbeeld, werd in Londen een nieuwe ambulancedienst ingevoerd. Deze mislukte totaal, omdat degenen die ermee moesten werken er een hekel aan hadden. Met de juiste mensen voor een project en de juiste communicatie creëer je brede ondersteuning (Eng. buy-in) en zijn dit soort situaties te voorkomen.

PT Embedded Systems: Requirements zijn volgens u onder te verdelen in verschillende typen en subtypen. Wat is bijvoorbeeld het verschil tussen functionele en niet-functionele requirements?

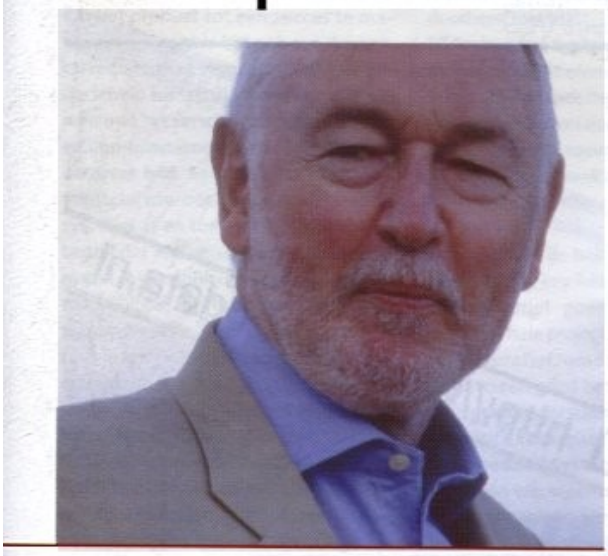
James Robertson: Er zijn allerlei categorieën, typen en subtypen die je kunt gebruiken bij het opstellen van een requirements-specificatie. Functionele requirements, waarbij het gaat om een fundamentele functionaliteit, beschrijven wat er moet worden gedaan. Ze zijn als werkwoorden te definiëren. De term niet-functionele requirements is enigszins onduidelijk, maar die requirements beschrijven een eigenschap, zoals bruikbaar (Eng. usable), ondersteunend (Eng. supportable) en te onderhouden (Eng. maintainable). Ook prestatie (Eng. performance) behoort hiertoe. Niet-functionele requirements worden helaas over het algemeen over het hoofd gezien, terwijl ze wel degelijk heel belangrijk zijn.

PT Embedded Systems: Een van de interessantste aspecten die u bespreekt, is het hergebruik van requirements.

James Robertson: De meeste organisaties maken producten die veel overeenstemming vertonen. Het komt zelden voor dat een bedrijf de ene dag een telefooncentrale ontwerpt en de volgende dag een bankapplicatie. Wat zo'n bedrijf wel maakt, is bijvoorbeeld een besturingssysteem voor een wasmachine en een besturingssysteem voor een telefoon schakelsysteem. De requirements daarvoor zullen uiteraard niet precies hetzelfde zijn, maar er zijn zeker

James Robertson benadrukt:

het specificeren



overeenkomsten op abstract niveau, die je kunt benutten.
Er zijn twee problemen bij hergebruik. Allereerst moet je dergelijke requirements kunnen vinden. Daar heb je bijvoorbeeld een database voor nodig. Daarnaast is het van essentieel belang dat er vertrouwen bestaat in de bestaande requirements die je wilt hergebruiken. Het is net als met code. Programmeurs zijn best bereid een bestaande code te hergebruiken, maar alleen als ze die code vertrouwen. Deze moet dan ook in de praktijk zijn nut hebben bewezen.

'Ingenieurs zijn heel goed in uitvinden; de kunst is om ze dingen te laten uitvinden die te verkopen zijn'

Schlumberger, een onderneming actief op het gebied van pompsystemen, heeft door al haar requirements-specificaties te categoriseren de afleveringstijden aanzienlijk weten terug te brengen. Een brandbare vloeistof moet anders verpompt worden dan een niet brandbare, maar er zijn ook veel overeenkomsten. Daar moet je gebruik van maken.

PT Embedded Systems: U spreekt, in een analogie met software-patterns, over requirement patterns. Houdt dit in dat ook de oplossingen voor herbruikbare requirements hetzelfde zijn?

James Robertson: Als dat mogelijk is, dan is dat zeker een bonus. Als je requirements aan het specificeren bent, moet je niet te veel naar de oplossingen kijken. Het maken van requirements duurt heel lang en is zeer foutgevoelig, veel meer dan programmatuur. De oorzaak van fouten ligt in de praktijk voor 50 tot 60% in de requirements en voor slechts 5% in de code. De rest van de fouten heeft als oorzaak configuratie en systeemintegratie of betreft fouten in het management.
Programmeurs kunnen over het algemeen heel goed coderen wat vereist (Eng. required) wordt. Daar ligt het probleem dan ook niet. De grootste winst zit in het verkorten van het requirements proces.

PT Embedded Systems: Requirements kun je testen voordat je een oplossing hebt. Hoe doe je dat?

James Robertson: Een requirement die je niet kunt testen is geen requirement! Testbaarheid is het eerste waar je naar moet kijken en dat betekent dat je altijd moet kwantificeren. Bruikbaarheid kun je bijvoorbeeld alleen maar testen als je een percentage aanvaardbare gebruikersfouten hebt vastgesteld of de tijdsduur die nieuwe gebruikers nodig hebben om een transactie af te ronden. Zodra je aan een requirement een getal kunt hangen, kun je hem testen.
Vervolgens krijg je vragen, zoals: 'Valt de requirement binnen het bereik van de toepassing?'. Elke ontwerpgroep heeft de neiging om allerlei toeters en bellen toe te voegen. 'Gold plating', oftewel 'scope creep' is iets waar je heel attent op moet zijn. Daarmee samen hangt de vraag hoe belangrijk een requirement is. Iedere ontwerper weet dat een requirement die niet als essentieel wordt gecategoriseerd, wegvalt. Daar moet je dus heel streng in zijn.
Je moet dan ook bijvoorbeeld niet alleen – op een schaal van 1 tot 5 – vragen hoe blij de klant zou zijn met een bepaalde toevoeging, maar ook hoe ontevreden hij zou zijn, als die requirement er niet in zou zitten. Als je bijvoorbeeld gebruikers van een webdienst vraagt of ze elke dag een Dilbert cartoon willen zien, dan zullen de meesten daar bevestigend op antwoorden. Als die strip daarentegen ontbreekt, dan zullen de meesten dat echter niet erg vinden. Deze benadering is oorspronkelijk afkomstig uit de marketingwereld. Ontevredenheid is dikwijls veel belangrijker dan tevredenheid. Je wilt niet zozeer ontdekken wat de klant wil hebben, maar waar hij voor wil betalen.

PT Embedded Systems: U gaat in uw boek ook in op 'use cases'.

James Robertson: Ivar Jacobson[4] heeft iedereen een grote dienst bewezen door het begrip 'use case' te introduceren, maar heeft hij die term jammer genoeg niet echt duidelijk gedefinieerd, zodat er in de praktijk nogal wat verwarring over bestaat.
Het is in ieder geval belangrijk om de zaak van buiten af te bekijken. Het gaat om de respons van het product of de software op een 'business event'. In het geval van een embedded systeem zal zo'n 'business event' meestal een fysisch signaal zijn, maar de essentie is dat het voor het systeem dat je wilt bouwen een externe gebeurtenis is. Vervolgens kan je beschrijven wat er gebeurt in een use case scenario.

PT Embedded Systems: Wat zijn de 'deliverables' van het requirements proces?

James Robertson: In de eerste plaats een 'business requirement specifica- ›

tion'. Daar staat in wat noodzakelijk is om het product tot een succes te maken. Vervolgens is het ook een 'technical requirements specification'. Het onderscheid tussen die twee valt niet samen met het verschil tussen functioneel en non-functioneel, waar ik het al eerder over had. Een technisch requirement, bijvoorbeeld, is dat een online systeem af en toe moet 'pingen'. Een bedrijf dat veel verschillende producten maakt in verschillende productfamilies, zoals Philips, zal verlangen dat de software binnen de 'inter product architectuur' past. Ook dit is een technisch requirement. Het onderscheid tussen 'business' en 'technical' is altijd vast te stellen, zij het dat dit soms heel moeilijk is. Buiten deze twee deliverables heb je natuurlijk ook te maken met allerlei documenten die kosten, hulpbronnen en risico's beschrijven, maar dat zijn in eigenlijke zin geen deliverables van het requirements proces. Voordat een requirement in een deliverable komt, moet hij overigens eerst de tests succesvol hebben afgerond. In ons requirements-proces, dat we Volere noemen, spreken we van een kwaliteitspoort waar de requirements doorheen moeten.

PT Embedded Systems: Is het requirements-proces incrementeel?

James Robertson: Iteratief, zou ik zeggen, incrementeel doet denken aan het steeds opnieuw toevoegen, terwijl je in de praktijk juist vaak dingen moet weglaten. Het is in ieder geval geen 'waterval' proces.

Je begint met de belangrijkste functionaliteit (Eng. key functionality). Als die namelijk niet goed is omschreven, heeft het hele project geen zin. Het proces is iteratief, omdat je het meestal niet meteen goed hebt. Je zult dan ook in de praktijk verschillende releases van de specificaties maken.

PT Embedded Systems: Tenslotte een misschien wat vreemde vraag. Waarom worden requirements altijd geformuleerd als: 'the software shall...' of 'the system shall...' ? Het is een (haast archaïsche) formulering die je nergens anders tegenkomt.

James Robertson: Iedereen is het er nu eenmaal over eens om dat zo te formuleren. Het mag in elk geval geen 'zou' zijn. Aan requirements moet echt worden voldaan en in dat opzicht is een gebiedende vorm wel op zijn plaats. Het requirements-proces is kostbaar, tijdrovend en moeizaam. Als je echter het verkeerde systeem bouwt of iets samenstelt waar de klant niet gelukkig mee is, dan is de prijs nog veel hoger. Het is met requirements net als met testen: niet doen kost altijd meer dan wel doen.. ■

Informatie

QA Systems International,
tel.: (033) 433 00 00, www.qa-systems.com

Literatuur

1 Suzanne Robertson, James Robertson, 'Mastering the Requirements Process', Addison-Wesley (1999).
2 www.systemsguild.com.
3 PT Embedded Systems, november 2000, p. 34-39.
4 PT Embedded Systems, maart 2002, p. 20-22.